Expert System Design

April 24, 2019

Restaurant Recommendations
CS152 Artificial Intelligence LBA
Frances Pak and Anna Pauxberger

1 Executive Summary

The following expert system lets users specify food preferences, such as cuisine, price and proximity, and gives a recommendation of restaurants that fulfill the preferences. The knowledge base contains restaurants and their attributes, as well as the rules that determine for example that a price can be either equal to or lower than the user specified. The user is asked for input one by one, and dynamically checks whether there are restaurants fulfilling the requirements. Given the user choices, it uses Prolog (PySwip library) to query the knowledge base and return the restaurants.

2 The Code

2.1 Knowledge Base

The Knowledge Base lists the restaurants and their attributes. Price ranges, distances, and ratings were found using Google Maps. Price ranges are categorized as low, medium, and high for simplicity of experience on the user's end. Ratings are categorized as okay, decent, good, and delicious for similar reasons. Distances are rounded to the nearest whole number.

2.2 Rules

Prolog goes through the following rules to determine whether the restaurants in the knowledge base satisfy the user's preferences. - **cuisine_type([Xh|Xt], Y) :- member(Xh,Y); cuisine_type(Xt, Y).** Iterative rule that goes through a list and determines if a chosen cuisine is in the list of specified cuisines for a restaurant. - **price_range(X, Y) :- X == Y.** Determines if the user-specified price preference is equal to what the restaurant has. - **max_distance(X, Y) :- Y =< X.** Ensures that the distance of the restaurant is smaller or equal to what the user specified. - **rating_at_least(X, Y) :- X =< Y.** Ensures that the rating of a restaurant is larger or equal to what the user specified.

```
In [1]: KB = """

        % Restaurants and their attributes
```

```
pick_restaurant(saigon) :- cuisine(Cuisine), cuisine_type(Cuisine, [vietnamese, vegetar

pick_restaurant(fasongsong) :- cuisine(Cuisine), cuisine_type(Cuisine, [korean, vegetar

pick_restaurant(chori) :- cuisine(Cuisine), cuisine_type(Cuisine, [argentinian]), price

pick_restaurant(cafe_ditali) :- cuisine(Cuisine), cuisine_type(Cuisine, [italian, arger

pick_restaurant(hong_kong_style) :- cuisine(Cuisine), cuisine_type(Cuisine, [chinese])

pick_restaurant(cabaña) :- cuisine(Cuisine), cuisine_type(Cuisine, [argentinian]), pric

pick_restaurant(empanairo) :- cuisine(Cuisine), cuisine_type(Cuisine, [argentinian]), p

pick_restaurant(artemisia) :- cuisine(Cuisine), cuisine_type(Cuisine, [european, vegan]

pick_restaurant(mcdonalds) :- cuisine(Cuisine), cuisine_type(Cuisine, [fastfood]), pric

pick_restaurant(biwon) :- cuisine(Cuisine), cuisine_type(Cuisine, [korean]), price(Pric

pick_restaurant(burger_king) :- cuisine(Cuisine), cuisine_type(Cuisine, [fastfood]), pr

pick_restaurant(la_parolaccia_trattoria) :- cuisine(Cuisine), cuisine_type(Cuisine, [it

pick_restaurant(baifu) :- cuisine(Cuisine), cuisine_type(Cuisine, [chinese]), price(Pri

pick_restaurant(loving_hut) :- cuisine(Cuisine), cuisine_type(Cuisine, [vegan]), price


% Rules

cuisine_type([Xh|Xt], Y) :- member(Xh,Y); cuisine_type(Xt, Y).
price_range(X, Y) :- X == Y.
max_distance(X, Y) :- Y =< X.
rating_at_least(X, Y) :- X =< Y.

"""

with open("KB.pl", "w") as text_file:
    text_file.write(KB)
```

2.3 Helper Functions

```
In [2]: def get_user_input():
            """This function asks the user for input about their cuisine, price, rating and di

            # USER CUISINE
            user_cuisine_int = raw_input("What type of food do you feel like? Type the number
```

```
try:
    # check for exit scenario
    if user_cuisine_int == "exit": return None, None, None, None
    # check that the input is a valid option
    if int(user_cuisine_int) not in [1,2,3,4,5,6,7,8,9]:
        print("Please enter a number between 1-9.")
        return get_user_input()
except ValueError:
    # exception if the input is not an integer
    print("Please enter a valid integer, e.g. 1 for Vietnamese food.")
    return get_user_input()
cuisine_dict = {"1":"vietnamese", "2":"vegetarian", "3":"korean", "4":"italian","5"
user_cuisine = cuisine_dict[user_cuisine_int]

# USER PRICE
user_price_int = raw_input("What is your price range? Type the number of your price
try:
    # check for exit scenario
    if user_price_int == "exit": return None, None, None, None
    # check that the input is a valid option
    if int(user_price_int) not in [1,2,3]:
        print("Please enter a number between 1-3.")
        return get_user_input()
except ValueError:
    # exception if the input is not an integer
    print("Please enter a valid integer, e.g. 1 for low price.")
    return get_user_input()
price_dict = {"1":"low", "2":"medium", "3":"high"}
user_price = price_dict[user_price_int]

# check temporary requirements with minimum requirements for other variables
temp = query_KB(user_cuisine, user_price, 1, 9)
if not temp:
    print("Please choose another combination. Unfortunately, this food is not avail
    return get_user_input()
else:
    print("Here are the potential options so far: {}".format((', '.join(temp))))


# USER RATING
user_rating = raw_input("How good does the food have to be, at minimum? Type the nu
try:
    # check for exit scenario
    if user_rating == "exit": return None, None, None, None
    # check that the input is a valid option
    if int(user_rating) not in [1,2,3,4]:
        print("Please enter a number between 1-4.")
        return get_user_input()
```

```
        except ValueError:
            # exception if the input is not an integer
            print("Please enter a valid integer, e.g. 1 for okay.")
            return get_user_input()

         # check temporary requirements with minimum requiremets for other variable
        temp = query_KB(user_cuisine, user_price, user_rating, 9)
        if not temp:
            print("Please choose another combination. Unfortunately, this food is not avail
            return get_user_input()
        else:
            print("Here are the potential options so far: {}".format((', '.join(temp))))

        # USER DISTANCE
        user_distance = raw_input("What should be the maximum distance (km) of the restaura

        # check that the input is an integer
        if not int(user_distance):
            print("Please enter a number.")
            return get_user_input()

        return user_cuisine, user_price, user_rating, user_distance


    def query_KB(user_cuisine, user_price, user_rating, user_distance):
        """Adsd the user input to the knowledge base, query the solution, removes the facts

        # assert (add) the user input to the knowledge base
        prolog.asserta("cuisine(" + str([user_cuisine]) + ")")
        prolog.asserta("price(" + str(user_price) + ")")
        prolog.asserta("rating(" + str(user_rating) + ")")
        prolog.asserta("distance(" + str(user_distance) + ")")

        # ensure solution uniqueness by defining a set
        solution_set = set()
        for solution in prolog.query("pick_restaurant(X)."):
            solution_set.add(solution.get("X", ""))

        # retract (remove) all facts from the database
        call(retractall(cuisine))
        call(retractall(price))
        call(retractall(rating))
        call(retractall(distance))

        return solution_set
```

2.4 Main Program & Test

```
In [3]: from pyswip.prolog import Prolog
        from pyswip.easy import *

        prolog = Prolog() # global handle to interpreter

        cuisine = Functor("cuisine",1)
        price = Functor("price",1)
        rating = Functor("rating",1)
        distance = Functor("distance",1)
        retractall = Functor("retractall")

        def main():
            prolog.consult("KB.pl") # open the KB
            pick_restaurant = Functor("pick_restaurant",1)
            prolog.dynamic("cuisine/1")
            prolog.dynamic("price/1")
            prolog.dynamic("rating/1")
            prolog.dynamic("distance/1")


            # get user's preferences via user input
            print("To exit the expert system, type 'exit'.")
            user_cuisine, user_price, user_rating, user_distance = get_user_input()

            restaurant_choice = query_KB(user_cuisine, user_price, user_rating, user_distance)

            if restaurant_choice:
                if len(restaurant_choice) == 1:
                    print("We recommend the following restaurant:")
                else:
                    print("We recommend the following restaurants:")
                for element in restaurant_choice:
                        print(element)
            else:
                print("We're sorry! It looks like there are no restaurants available with these
```

2.4.1 First successful test case (Anna's choice)

```
In [4]: # prolog = Prolog()
        main() # 7,2,3,10
```

```
To exit the expert system, type 'exit'.
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
How good does the food have to be, at minimum? Type the number of your preference. 1-okay, 2-de
What should be the maximum distance (km) of the restaurant? Round to the nearest integer. 10We
artemisia
```

2.4.2 Second successful test case (Frances' choice)

```
In [5]: #prolog = Prolog()
        main() #3, 2, 4, 3

To exit the expert system, type 'exit'.
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
How good does the food have to be, at minimum? Type the number of your preference. 1-okay, 2-de
What should be the maximum distance (km) of the restaurant? Round to the nearest integer. 3We r
fasongsong
```

2.4.3 First failure test case (cuisine)

```
In [6]: # Test case: string input, out of range input, exit on cuisine level
        # 'no', 10, 'exit'
        prolog = Prolog()
        main()

To exit the expert system, type 'exit'.
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
```

2.4.4 First failure test case (price range)

```
In [7]: # Test case: string input, out of range input, exit on price range level
        # 1, 'high', 1, 4, exit
        prolog = Prolog()
        main()

To exit the expert system, type 'exit'.
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
What type of food do you feel like? Type the number of your preferred cuisine. 1-Vietnamese, 2-
```

3 Appendix

3.1 Member contributions

Data was collected together. The prolog code was done by Frances. The main skeleton of the PySWIP code was done by Anna. Other parts of the python code (i.e. error check, inputs, returning temporary restaurants, etc) and the write-up was done together.

3.2 Noteworthy HCs

-

4 audience - Minerva students don't have capacity to spend a lot of time on searching for the right place. We picked the best places in the surrounding, give simple options, and make it easy for our audience to find the right place quickly. As soon as their preferences result in no choices, we notify the student and ask them to submit another request so that they don't waste time continuing to ask questions when we already know that there is no place satisfying their preferences. By using numbers as input, students don't have to worry about spelling things correctly, making their user experience as easy and smooth as possible. (We hope you, Professor, as your secondary if not primary audience, also enjoyed the process.)

4.1 References

Referenced code from class session 12.2 where PySWIP was used.

4.2 Attachments

The following table shows the collected data, also available here.

	A	B	C	D	E	F
1	**Restaurant**	**Type**	**Price Range (ARS)**	**Walking Distance (km)**	**Google Rating**	
2	Saigon	Vietnamese, vegetari	Low	0.13	4.5	
3	Fo Song Song	Korean, vegetarian	Medium	0.076	4.5	
4	Chori	Argentinian	Low	5.3	4.4	
5	Cafe Ditali	Italian, Argentinian	Low	0.18	4	
6	Hong Kong Style	Chinese	Medium	8.7	4.1	
7	Cabaña Las Lilas	Argentinian	High	1.8	4.5	
8	Empanairo	Argentinian	Low	0.07	4.4	
9	Artemisia	European, vegan	Medium	6.4	4.4	
10	McDonalds	fast food	low	0.75	3.9	
11	Bi Won	Korean	Medium	2.2	4.1	
12	Burger King	fast food	low	3.3	3.8	
13	La Parolaccia Trattoria	Italian	high	1.7	4.4	
14	Baifu	chinese	Medium	6.1	4	
15	Loving Hut	vegan	Medium	5.2	4.3	
16						
17						
18	**Price range conversion**		**Rating conversion**		**Walking distance**	
19	Low	< 200	okay	>3.75	rounded to nearest whole number	
20	Medium	< 400	decent	>4		
21	HIgh	> 400	good	>4.25		
22			delicious	>4.5		


Empanairo
CAFÉ +
2 MEDIALUNAS
$65
HAPPY
HOUR
PROMOS
3 EMPANADAS
+CERVEZA $170
+BEBIDA $110
PIZZETAS
MOZZARELLA
FUGAZZETA